Hilfsmittel

Beliebige schriftliche Unterlagen; Internet

Punktzahl

-

Zeit

-

Anweisung

* Sie können Ihre Lösungen entweder ins elektronische Prüfungsdokument oder in die gedruckte Prüfung (z.B. für Grafiken) notieren.
* Für die gedruckte Prüfung gilt:
  + Falls Sie auf Falls Sie Zusatzblätter brauchen, benutzen Sie für jede Aufgabe separate Zusatzblätter.
  + Sie können auch mit Bleistift schreiben.
  + Benutzen Sie keinen roten Stift.

Abgabe

* Laden Sie das Word-Dokument (und eventuelle Anhänge) in den Lernraum hoch.

1. **JavaScript – 4 gewinnt**. In JavaScript soll das Spiel 4 gewinnt (engl. connect 4) implementiert werden. Dabei werden Spielsteine (engl. discs) in ein Spielfeld (engl. field) eingeworfen.

Für die Spielsteine werden folgende Konstanten definiert:

// discs

var DISC\_YELLOW = "y";

var DISC\_RED = "r";

var NO\_DISC = "\_";

Das Spielfeld besteht üblicherweise aus 7 Spalten und 6 Reihen. Dafür soll in JavaScript ein Array genutzt werden, welches wiederum Arrays als Elemente enthält.

// create field

var field = new Array(7);

for (var column = 0; column < 7; column++) {

field[column] = new Array(6);

}

Die Methode toString kann genutzt werden, um ein Spielfeld als String darzustellen.

/\*\*

\* Create a string representation of the field.

\*

\* @param field the field

\* @returns {String} the string representation.

\*/

function toString(field) {

var result = "";

for (var rowNo = 0; rowNo < field[0].length; rowNo++) {

for (var columnNo = 0; columnNo < field.length; columnNo++) {

var disc = field[columnNo][rowNo];

result += disc + " ";

}

result += "\n";

}

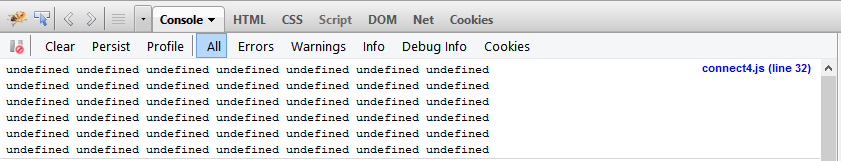
return result;

}

Die Ausgabe von

console.log(toString(field));

in der Konsole von Firebug ist



* 1. **Spielfeld initialisieren**. Das Spielfeld soll mit den Werten NO\_DISC initialisiert werden. Schreiben Sie eine Funktion init, welche das gegebene Spielfeld initialisiert.

/\*\*

\* Init field.

\*

\* @param field the field

\*/

Der Aufruf von folgendem Codeausschnitt (inklusive den vorhergehenden Codeausschnitten)

init(field);

console.log(toString(field));

zeigt in der Konsole von Firebug



* 1. **Spielstein ablegen**. Schreiben Sie eine Funktion drop, welche einen Spielstein ins Spielfeld ablegt.

/\*\*

\* Drop the disc to specified column.

\*

\* @param disc the disc; must be either DISC\_YELLOW or DISC\_RED

\* @param {Integer} columnNo the column number

\* @return {Boolean} returns true on success

\*/

Der Aufruf von folgendem Codeausschnitt (inklusive den vorhergehenden Codeausschnitten)

drop(DISC\_YELLOW, 4);

drop(DISC\_RED, 3);

drop(DISC\_YELLOW, 4);

console.log(toString(field));

zeigt in der Konsole von Firebug



* 1. **Anzahl Steine in einer Spalte (funktionale Programmierung)**. Schreiben Sie eine Funktion countDiscsInColumn, welche die Anzahl Spielsteine in einer Spalte zählt.

Nutzen Sie die Methoden höherer Ordnung map und reduce von Arrays.

/\*\*

\* Count the number of discs in the given column.

\*

\* @param field the field

\* @param {Integer} columnNo the column

\* @returns {Integer} the number of discs

\*/

function countDiscsInColumn(field, columnNo) {

var column = field[columnNo];

// Teil 1: map

var column01 = column.map( );

// Teil 2: reduce

return column01.reduce( );

}

1. **JavaScript – 4 gewinnt II**. Die Lösung aus Aufgabe 1 entspricht nicht den best practices.

Schreiben Sie ihre Lösung von Aufgabe 1 so um, dass Sie objekt-orientiert wie folgt verwendet werden kann. Ihre Lösung darf den globalen Namensraum um maximal 1 Element bereichern.

var field = new Field(5,5);

console.log(field.toString());

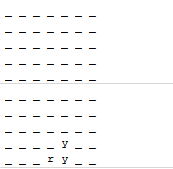
field.drop(Field.DISC\_YELLOW, 4);

field.drop(Field.DISC\_RED, 3);

field.drop(Field.DISC\_YELLOW, 4);

console.log(field.toString());

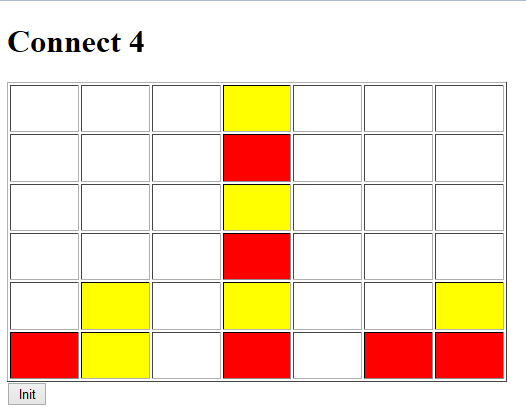
In der Firebug Konsole soll folgendes angezeigt werden:



1. **JavaScript – 4 gewinnt III**. In Aufgabe 2 haben Sie das Spiel 4 gewinnt in JavaScript implementiert. Als nächstes soll das Spiel ins Web.

Die Web-Seite soll wie folgt aussehen. Funktionalität:

* Klick auf eine Spalte fügt einen neuen Spielstein hinzu.
* Klick auf Init löscht das Spielfeld.



Gegen ist die Datei connect4.js. Die Datei enthält zusätzlich die Klasse FieldAsTable. Nutzen Sie die Klasse, um das Spiel zu implementieren.

Fügen Sie den fehlenden Code an die markierte Stelle in der Datei connect4.html ein.

* 1. **Anzeige des Spielfeldes.** Das Spielfeld soll mittels der Klasse zu Beginn FieldAsTable hinzugefügt werden. Die Funktion createElement kann verwendet werden, um ein <tbody>-Element zu erzeugen. Fügen Sie dieses an der passenden Stelle ins HTML-Dokument ein.

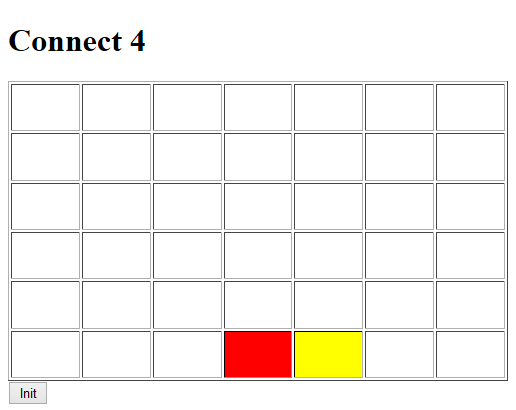
Prüfen Sie ihre Implementierung, in dem Sie testweise Steine hinzufügen. Folgender Code

var field = new Field(7,6);

field.drop(Field.DISC\_YELLOW, 4);

field.drop(Field.DISC\_RED, 3);

sollte beispielsweise folgende Darstellung erzeugen. Vergessen Sie nicht, mittels FieldAsTable.prototype.update die Darstellung zu aktualisieren.



* 1. **Schaltfläche Init.** Fügen der Schaltfläche Init einen Click-Handler hinzu, der mittels der Funktion Field.prototype.init die Spielfläche frisch initialisiert und danach mittels FieldAsTable.prototype.update die Darstellung aktualisiert.
  2. **Spielzug**. Wenn in eine Spalte geklickt wird, soll in dieser Spalte ein Stein abgelegt werden. Nutzen sie die Methode FieldAsTable.prototype.addHandler um einen Event-Handler auf die <td>-Elemente zu registrieren.

Hinweis:

* Unnötige Verschmutzungen des globalen Namensraumes ergeben Abzug!

1. **ECMA-Referenz**. Gegeben ist folgender Codeausschnitt:

// Teilaufgabe a)

var x = 1 / 0;

console.log(x);

// Teilaufgabe b)

var y = x + 1;

console.log(y);

Der Code erzeugt folgende Ausgabe:

Infinity

Infinity

Erläutern Sie, wie diese Resultate zustande kommen unter Verweis auf die entsprechenden Regeln der ECMA-Script-Referenz (<http://www.ecma-international.org/ecma-262/5.1/>).

Geben Sie die genaue Kapitelnummer der Referenz sowie den massgebenden Unterpunkt an:

* 1. var x = 1 / 0;
  2. var y = x + 1;